

Trabalho Final

1 Introdução

O objetivo é desenvolver em VHDL um processador digital simples segundo as especificações contidas neste documento. Deve ser entregue os códigos desenvolvidos, um relatório e apresentar o processador digital funcionando no kit de FPGA do laboratório.

2 Especificação

2.1 Barramentos

Na arquitetura proposta, utilizaremos três barramentos principais, conforme ilustrado na imagem anexa:

- Barramento de Endereço: Indica o endereço que será acessado para leitura ou escrita, seja na memória, entrada ou saída.
- Barramento de Dados: Transporta os dados a serem lidos ou escritos no endereço apontado.
- Barramento de Controle: Transporta os sinais de controle, como:
 - Read: Indica que será realizada uma leitura de memória (ativa apenas se Memory Enable estiver ativa).
 - Write: Indica que será realizada uma escrita de memória (ativa apenas se Memory Enable estiver ativa).
 - Memory Enable: Habilita a leitura ou escrita na memória.
 - Input Enable: Habilita a leitura dos dispositivos de entrada.
 - Output Enable: Habilita a escrita nos dispositivos de saída.
 - Clock: Controle de ciclo do processador
 - Reset: reset para o estado inicial.

A conexão entre os barramentos, dispositivos e CPU pode ser visualizado na imagem abaixo.

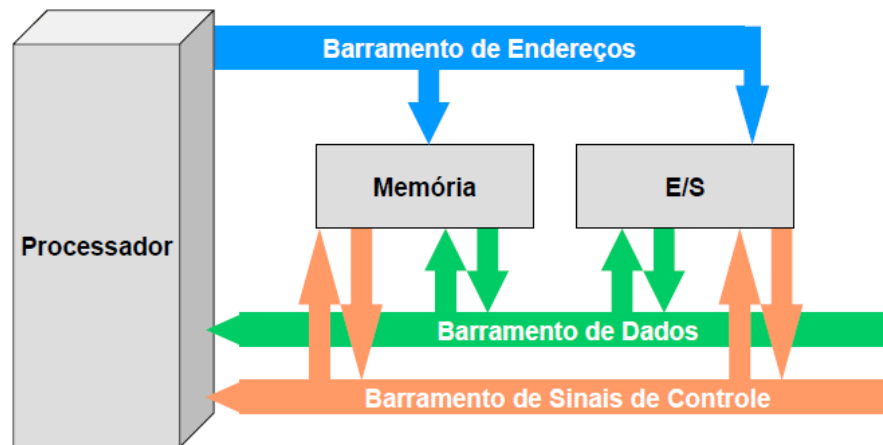


Figure 1: Barramentos

2.2 Componentes Internos da CPU

A CPU é composta pelos seguintes módulos:

- Input Unit: Utiliza 8 chaves do kit FPGA como entrada.
- Output Unit: Exibe dados de saída em 8 LEDs do kit FPGA.
- Memory Unit: Memória de 8 bits, utilizando módulos desenvolvidos em atividades anteriores.
- CPU:
 - ALU: Realiza operações lógicas e aritméticas (soma, subtração, AND, OR, NOT).
 - Control Unit:
 - * Decodificação de Instruções: Identifica a instrução e ativa os componentes necessários (ALU, registradores, etc.).
 - * Coordenação de Operações: Controla o fluxo de dados entre CPU, memória, e unidades de entrada e saída.
 - * Sinais de Controle: Gera sinais para sincronizar as operações entre ALU, memória, entrada e saída.

2.3 Registradores da CPU

A CPU terá os seguintes registradores:

- Flags (1 bit): Zero, Sinal, Carry, Overflow.

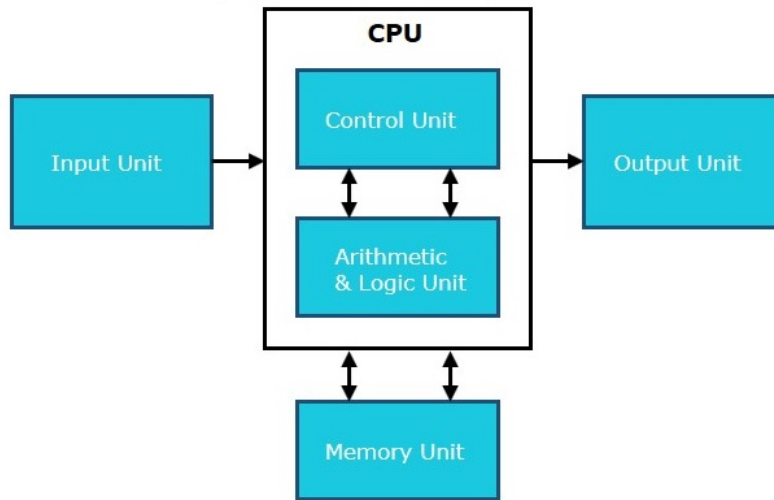


Figure 2: Componentes internos da CPU

- A e B (8 bits): Registradores de propósito geral.
- R (8 bits): Registrador para armazenar resultados da ALU.
- Program Counter (PC) (8 bits): Contador de instruções.
- Instruction Register (IR) (8 bits): Armazena a instrução atual.

2.4 Instruções

As instruções planejadas incluem operações aritméticas, lógicas e de controle de fluxo:

- ADD Reg1, Reg2: Soma valores de Reg1 e Reg2, armazena o resultado em R.
- SUB Reg1, Reg2: Subtrai Reg2 de Reg1, armazena o resultado em R.
- AND Reg1, Reg2: Realiza operação AND entre Reg1 e Reg2, armazena o resultado em R.
- OR Reg1, Reg2: Realiza operação OR entre Reg1 e Reg2, armazena o resultado em R.
- NOT Reg1: Inverte os bits de Reg1, armazena o resultado em R.
- CMP Reg1, Reg2: Compara Reg1 e Reg2, atualizando os registradores de flags.

- JMP addr: Salta para o endereço addr.
- JEQ addr: Salta para addr se o último resultado comparado for igual.
- JGR addr: Salta para addr se o último resultado comparado for maior.
- LOAD Reg1, addr: Carrega o valor do endereço addr para Reg1.
- STORE Reg1, addr: Armazena o valor de Reg1 no endereço addr.
- MOV Reg1, Reg2: Move o valor de Reg2 para Reg1.
- IN Reg1: Lê o valor das chaves do FPGA e armazena em Reg1.
- OUT Reg1: Exibe o valor de Reg1 nos LEDs.
- WAIT: (Possível inclusão) Aguarda um sinal de entrada para continuar

Note que as instruções como ADD podem usar os Registradores A,B, R e também podem receber números de 8 bits. Exemplos:

```

1 ADD A B      ; Soma A e B e Armazena em R (R=A+B)
2 ADD A R      ; Soma A e R e Armazena em R (R=A+R)
3 ADD B R
4 ADD R R      ;
5 ADD A 255    ; Soma A e 255 e Armazena em R (R=A+255)

```

A instruções STORE podem armazenar valores de um Registrador no endereço especificado. O endereço pode ser um número de 8 bits ou um Registrador. A instrução LOAD carrega em um registrador o valor de um endereço na memória que pode ser um número de 8 bits ou um registrador. Exemplos

```

1 STORE A 00000000
2 STORE R 00000010
3 STORE B R
4 LOAD A B
5 LOAD A 00000000

```

A implementação das instruções e codificação fica em aberto para implementar da forma que acharem mais conveniente.

3 Exemplos de programas para CPU

Os três exemplos a seguir servem para ajudar a entender como as instruções da CPU são utilizadas.

3.1 Soma

```
1 IN A ; Lê o que está no input para Registrador A
2 IN B ; Lê o que está no input para Registrador B
3 ADD A, B ; Realiza operação de soma de A+B armazenadno em R
4 OUT R ; Coloca o valor de R na saída
5 WAIT
```

3.2 Multiplicação

```
1 IN A ; Lê o que está no input para Registrador A
2 IN B ; Lê o que está no input para Registrador B
3 MOV R, 0 ; Coloca o valor zero no Registrador R
4 LOOP_START:
5     CMP B, 0 ; Compara B com 0
6     JEQ END_LOOP ; Se B for igual a 0, termina o loop
7     ADD R, A ; Soma A ao valor acumulado em R (R=R+A)
8     STORE R 255 ; Armazena o valor de R no endereço 255
9     SUB B, 1 ; Decrementa o multiplicador B (R=B-1)
10    MOV B, R ; Armazena o valor de R em B
11    LOAD R, 255 ; Carrega o valor do endereço 255 em R
12    JMP LOOP_START ; Repete o loop
13 END_LOOP:
14     OUT R
15 WAIT
```

3.3 Divisão

```
1 IN A ; Lê as entradas em A
2 IN B ; Lê as entradas em B
3 MOV R 0; Armazena 0 em R
4 STORE R 255 ; Armazena R no endereço 255
5 MOV R A ; Armazena o valor de A em R(R=A)
6 LOOP_START
7     SUB R B ; R=R-B
8     MOV A R ; A=R
9     LOAD R 255; R=ADDR(255)
10    ADD R 1; R=R+1
11    STORE R 255; ADDR(255)=R
12    MOV R A ; R=A
13    CMP R B;
14    JGR LOOP_START; R>B? Vai para LOOP_START
15 LOAD R 255; R=ADDR(255)
16 OUT R; Coloca na saída o valor de R
17 WAIT
```